An Improved CNF Encoding Scheme for Probabilistic Inference

Jean-Marie Lagniez

CRIL, U. Artois & CNRS
France

Overview

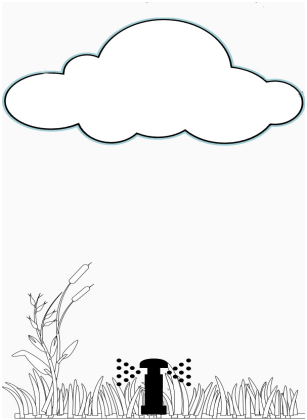
Probabilistic Graphical Models

Bayesian Network

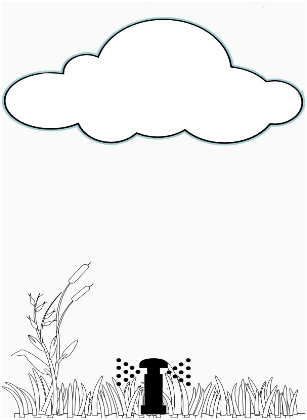
Weighted Model Counting

Bayesian Network Translations

Probabilistic Inference

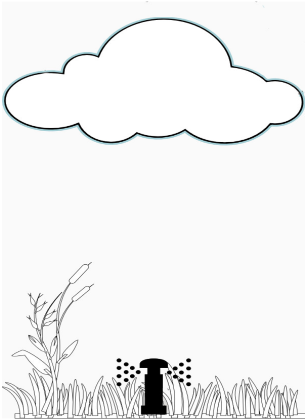


Probabilistic Inference



- Observations:
  - It is cloudy
  - The grass is wet
  - ...

Probabilistic Inference

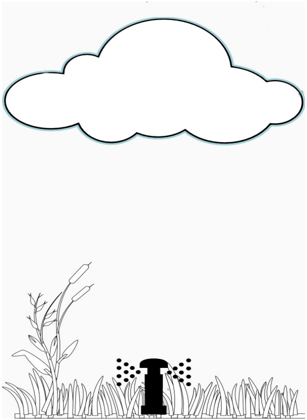


- Observations:
  - It is cloudy
  - The grass is wet
  - ...

day	cloudy	wet grass
1	1	1
2	1	1
3	1	0
4	0	0

$\mathbb{P}(\text{wet grass} \mid \text{cloudy}) = \frac{2}{3}$

Probabilistic Inference



- Observations:
  - It is cloudy
  - The grass is wet
  - ...

day	cloudy	wet grass
1	1	1
2	1	1
3	1	0
4	0	0

$$\mathbb{P}(\text{wet grass} \mid \text{cloudy}) = \frac{2}{3}$$

- Modelling a problem using a **joint distribution** on a set of random variables

Probabilistic Inference

- An **instantiation** of $\mathbf{X}$ assigns each $X \in \mathbf{X}$ of some value in the domain (finite) of X
- A **joint distribution** over $\mathbf{X}$ is a function $\mathbb{P}$ that maps each instantiation of $\mathbf{X}$ to $[0, 1]$
- **Evidence** is similar to instantiation but only some of the variables of $\mathbf{X}$ are assigned

Probabilistic Inference: example

Cloudy	Rain	Sprinkler	Wet	$\mathbb{P}$
0	0	0	0	0.225
0	0	0	1	0
0	0	1	0	0.45
0	0	1	1	0.18
0	1	0	0	0.0025
0	1	0	1	0.0225
0	1	1	0	0
0	1	1	1	0.025
1	0	0	0	0.09
1	0	0	1	0
1	0	1	0	0.002
1	0	1	1	0.008
1	1	0	0	0.036
1	1	0	1	0.324
1	1	1	0	0
1	1	1	1	0.04

Probabilistic Inference: example

Cloudy	Rain	Sprinkler	Wet	$\mathbb{P}$
0	0	0	0	0.225
0	0	0	1	0
0	0	1	0	0.45
0	0	1	1	0.18
0	1	0	0	0.0025
0	1	0	1	0.0225
0	1	1	0	0
0	1	1	1	0.025
1	0	0	0	0.09
1	0	0	1	0
1	0	1	0	0.002
1	0	1	1	0.008
1	1	0	0	0.036
1	1	0	1	0.324
1	1	1	0	0
1	1	1	1	0.04

- probability of instantiation w

$$\mathbb{P}(C = 0, R = 1, S = 0, W = 1) = \mathbb{P}(C_0, R_1, S_0, W_1) = 0.0225$$

Probabilistic Inference: example

Cloudy	Rain	Sprinkler	Wet	$\mathbb{P}$
0	0	0	0	0.225
0	0	0	1	0
0	0	1	0	0.45
0	0	1	1	0.18
0	1	0	0	0.0025
0	1	0	1	0.0225
0	1	1	0	0
0	1	1	1	0.025
1	0	0	0	0.09
1	0	0	1	0
1	0	1	0	0.002
1	0	1	1	0.008
1	1	0	0	0.036
1	1	0	1	0.324
1	1	1	0	0
1	1	1	1	0.04

- probability of instantiation w

$$\mathbb{P}(C=0, R=1, S=0, W=1) = \mathbb{P}(C_0, R_1, S_0, W_1) = 0.0225$$

- probability of evidence e

$$\mathbb{P}(S_0, W_1) = 0 + 0.0225 + 0 + 0.324 = 0.3465$$

Probabilistic Inference: example

Cloudy	Rain	Sprinkler	Wet	$\mathbb{P}$
0	0	0	0	0.225
0	0	0	1	0
0	0	1	0	0.45
0	0	1	1	0.18
0	1	0	0	0.0025
0	1	0	1	0.0225
0	1	1	0	0
0	1	1	1	0.025
1	0	0	0	0.09
1	0	0	1	0
1	0	1	0	0.002
1	0	1	1	0.008
1	1	0	0	0.036
1	1	0	1	0.324
1	1	1	0	0
1	1	1	1	0.04

- The size of the joint distribution is exponential in the number of variables!

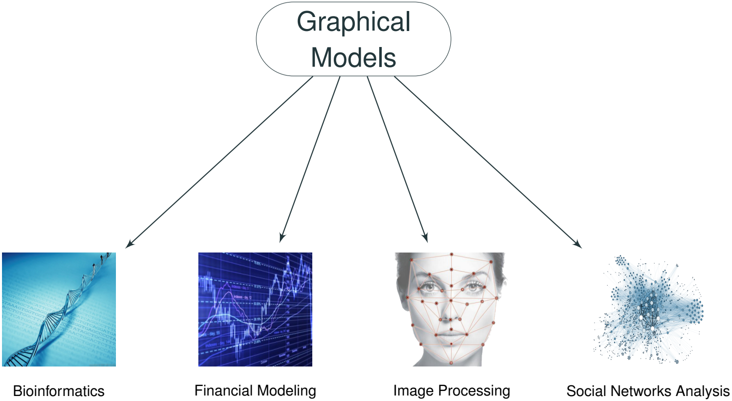
- probability of instantiation *w*

$$\mathbb{P}(C=0, R=1, S=0, W=1) = \mathbb{P}(C_0, R_1, S_0, W_1) = 0.0225$$

- probability of evidence *e*

$$\mathbb{P}(S_0, W_1) = 0 + 0.0225 + 0 + 0.324 = 0.3465$$

Graphical Models



Graphical models are a well-studied framework for representing high-dimensional probability distributions, with a wide spectrum of applications.

Overview

Probabilistic Graphical Models

Bayesian Network

Weighted Model Counting

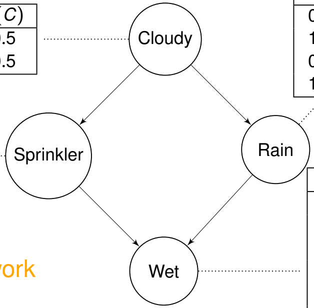
Bayesian Network Translations

Graphical Models

C	$\mathbb{P}(C)$
0	0.5
1	0.5

R	C	$\mathbb{P}(R \mid C)$
0	0	0.9
1	0	0.1
0	1	0.2
1	1	0.8

S	C	$\mathbb{P}(S \mid C)$
0	0	0.5
1	0	0.5
0	1	0.9
1	1	0.1

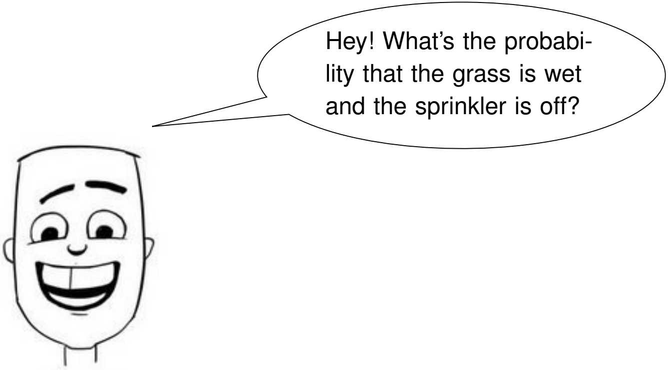


W	S	R	$\mathbb{P}(W \mid S, R)$
0	0	0	1.0
1	0	0	0.0
0	0	1	0.1
1	0	1	0.9
0	1	0	0.2
1	1	0	0.8
0	1	1	0.0
1	1	1	1.0

A Bayesian Network

- Graphical models are commonly separated into:
  - **directed** models (a.k.a Bayesian networks), with conditional probability tables (CPTs) over a directed (acyclic) graph,
  - **undirected** models (a.k.a. Markov networks), with energy functions over the cliques of an undirected graph.

Bayesian network

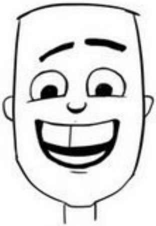


The semantics of Bayesian networks implies the following joint distribution:

$$\mathbb{P}(x_1, x_2, \ldots, x_n) = \prod_i \mathbb{P}(x_i | u_i)$$

Bayesian network

Hey! What's the probability that the grass is wet and the sprinkler is off?

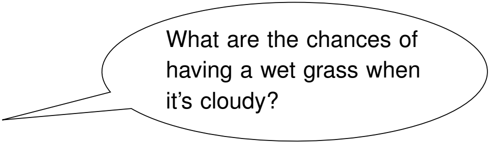


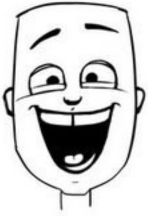
$$\begin{aligned}\mathbb{P}(c_0, r_1, s_0, w_1) &= \mathbb{P}(c_0) \times \mathbb{P}(r_1|c_0) \times \mathbb{P}(s_0|c_0) \times \mathbb{P}(w_1|s_0, r_1)\\ &= 0.5 \times 0.1 \times 0.5 \times 0.9\\ &= 0.0225\end{aligned}$$

The semantics of Bayesian networks implies the following joint distribution:

$$\mathbb{P}(x_1, x_2, \ldots, x_n) = \prod_i \mathbb{P}(x_i|u_i)$$

Probabilistic Inference





Product rule: $\mathbb{P}(A,X) = \mathbb{P}(A|X) \times \mathbb{P}(X)$

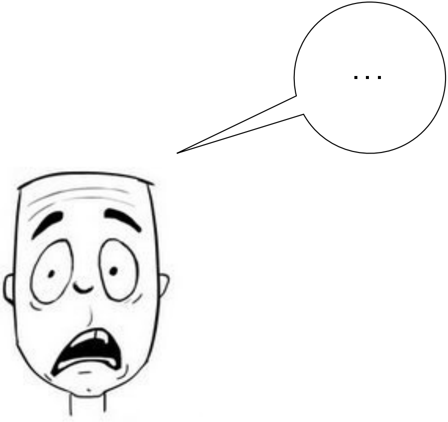
$$\mathbb{P}(w_1 \mid c_1) = \frac{\mathbb{P}(w_1,c_1)}{\mathbb{P}(c_1)}$$

Bayes' rule: $\mathbb{P}(A|X) = \frac{\mathbb{P}(X|A)\times\mathbb{P}(A)}{\mathbb{P}(X)}$

$$\mathbb{P}(c_1 \mid s_1) = \frac{\mathbb{P}(s_1,c_1)\times\mathbb{P}(s_1)}{\mathbb{P}(c_1)}$$

- Probabilistic inference is a basic task for handling more complex queries:
  - Conditional probabilities: evaluating the probability of an event $\boldsymbol{y}$ given an evidence $\boldsymbol{x}$
  - Most probable explanations: given an evidence $\boldsymbol{x}$, find an assignment $\boldsymbol{y}$ over the complementary variables that maximizes $\mathbb{P}(\boldsymbol{y} \mid \boldsymbol{x})$
  - ...

Graphical Models



- Unfortunately, probabilistic inference is $\#$**P-complete** in graphical models, including both Bayesian networks and Markov networks.

Weighted Model Counting

Yet, in AI, a great deal of effort has been spent in solving instances of the $\#$**SAT** problem, which is to count the number of models of an input CNF formula F.

- **Search-based methods:** Cachet, SharpSAT, etc.
- **Compilation-based methods:** C2D, SDD, ..., targetting a class of Boolean circuits (ex: DNNF) for which model counting is solvable in polynomial time

Weighted Model Counting

Yet, in AI, a great deal of effort has been spent in solving instances of the **$\#$SAT** problem, which is to count the number of models of an input CNF formula F.

- **Search-based methods:** Cachet, SharpSAT, etc.
- **Compilation-based methods:** C2D, SDD, ..., targetting a class of Boolean circuits (ex: DNNF) for which model counting is solvable in polynomial time

Most of these methods can handle **weighted** $\#$SAT instances, in which the literals of the input formula are associated with weights.

Overview

Probabilistic Graphical Models

Bayesian Network

Weighted Model Counting

Bayesian Network Translations

Weighted CNF Formula

- Let $V = \{v_1, \cdots, v_n\}$ be a set of Boolean variables.
- Let $L = \{v_i, \overline{v}_i \mid v_i \in V\}$ be the set of literals over V.
- A **wCNF formula** over V is a pair (F, w) such that:
  - F is a conjunction of clauses,
  - w is a map from L to $\mathbb{Q}$ (i.e. a weighting of literals)

Weighted Model Counting

- The weight of a variable assignment $\boldsymbol{v} \in \{0,1\}^n$ is the product of weights of literals which are true in $\boldsymbol{v}$:

$$W(\boldsymbol{v}) = \prod_{\ell \in L, \boldsymbol{v} \models \ell} w(\ell)$$

Weighted Model Counting

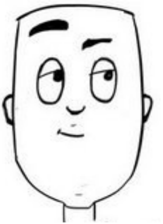
- The weight of a variable assignment $\boldsymbol{v} \in \{0,1\}^n$ is the product of weights of literals which are true in $\boldsymbol{v}$:

$$W(\boldsymbol{v}) = \prod_{\ell \in L, \boldsymbol{v} \models \ell} w(\ell)$$

- The weighted model count of a wCNF formula is the sum of weights of the models of F:

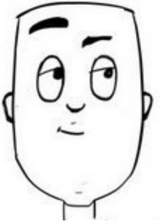
$$W(F) = \sum_{\boldsymbol{v} \in \{0,1\}^n, \boldsymbol{v} \models F} W(\boldsymbol{v})$$

Weighted Model Counting



So . . . Can we **reduce** the problem of probabilistic inference to Weighted Model Counting?

Weighted Model Counting



So . . . Can we **reduce** the problem of probabilistic inference to Weighted Model Counting?

Yes! The idea is to find a **translation function** τ mapping

- any graphical model (Bayes or Markov net) N to a weighted CNF formula $\tau(N)$, and
- any partial assignment $\boldsymbol{x}$ to a term $\tau(\boldsymbol{x})$,

such that

$$\mathbb{P}_N(\boldsymbol{x}) = \frac{W[\tau(N) \wedge \tau(x)]}{W[\tau(N)]}$$

where $W[\tau(N)]$ is the **partition constant** (= 1 for BNs).

Overview

Probabilistic Graphical Models

Bayesian Network

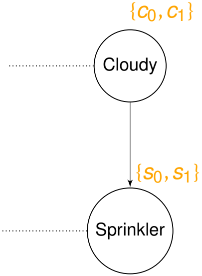
Weighted Model Counting

Bayesian Network Translations

ENC1 [Darwiche, KR'02]

C	$\mathbb{P}(C)$
0	0.5
1	0.5

S	C	$\mathbb{P}(S \mid C)$
0	0	0.5
1	0	0.5
0	1	0.9
1	1	0.1

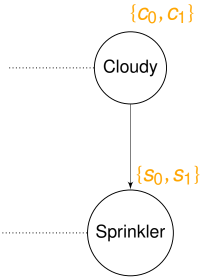


- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$

ENC1 [Darwiche, KR'02]

C	$\mathbb{P}(C)$
0	0.5
1	0.5

S	C	$\mathbb{P}(S \mid C)$
0	0	0.5
1	0	0.5
0	1	0.9
1	1	0.1



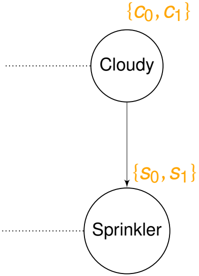
$c_0 \vee c_1$
$\overline{c_0} \vee \overline{c_1}$

$s_0 \vee s_1$
$\overline{s_0} \vee \overline{s_1}$

- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$
- A set of indicator clauses for each random variable X_i (direct encoding)

ENC1 [Darwiche, KR'02]

C	$\mathbb{P}(C)$	
0	0.5	θ_{c1}
1	0.5	θ_{c2}



$c_0 \vee c_1$
$\overline{c_0} \vee \overline{c_1}$

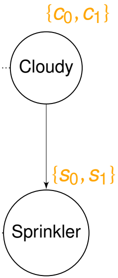
S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	θ_{s1}
1	0	0.5	θ_{s2}
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}

$s_0 \vee s_1$
$\overline{s_0} \vee \overline{s_1}$

- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$
- A set of indicator clauses for each random variable X_i (direct encoding)
- A parameter variable θ_{ir} for each row r in the table of X_i, and a weight $w(\theta_{ir}) = \mathbb{P}(x_i \mid \boldsymbol{x}_r)$

$c_0 \leftrightarrow \theta_{c1}$
$c_1 \leftrightarrow \theta_{c2}$

C	$\mathbb{P}(C)$	
0	0.5	θ_{c1}
1	0.5	θ_{c2}



$c_0 \vee c_1$
$\overline{c_0} \vee \overline{c_1}$

$s_0 \wedge c_0 \leftrightarrow \theta_{s1}$
$s_1 \wedge c_0 \leftrightarrow \theta_{s2}$
$s_0 \wedge c_1 \leftrightarrow \theta_{s3}$
$s_1 \wedge c_1 \leftrightarrow \theta_{s4}$

S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	θ_{s1}
1	0	0.5	θ_{s2}
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}

$s_0 \vee s_1$
$\overline{s_0} \vee \overline{s_1}$

- An indicator variable v_{ij} for each random variable X_i and domain value $j \in D(X_i)$
- A set of indicator clauses for each random variable X_i (direct encoding)
- A parameter variable θ_{ir} for each row r in the table of X_i, and a weight $w(\theta_{ir}) = \mathbb{P}(x_i \mid \boldsymbol{x}_r)$
- A set of equivalences describing the semantics of each row in the table of X_i

Local Structure

- In practice, **variables are not** necessary **binary** and **CPTs** can be quite **large**

A	B	C	$\mathbb{P}(c \mid a, b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- Thus, the generated **CNF encoding** can be **large**, challenging state-of-the-art weighted model counters

Remove Inconsistent Assignments

A	B	C	$\mathbb{P}(c \mid a, b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

Remove Inconsistent Assignments

A	B	C	$\mathbb{P}(c \mid a, b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- The weight of a model is obtained by multiplying the weights of its positive parameters
- Then, all models which contain a positive parameter with a weight of 0 can be removed

Remove Inconsistent Assignments

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- The weight of a model is obtained by multiplying the weights of its positive parameters
- Then, all models which contain a positive parameter with a weight of 0 can be removed

$$\neg a_1 \vee \neg b_1 \vee \neg c_1 \quad \neg a_2 \vee \neg b_1 \vee \neg c_1 \quad \neg a_2 \vee \neg b_1 \vee \neg c_2$$

Equal Parameters

A	B	C	$\mathbb{P}(c \mid a, b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

Equal Parameters

A	B	C	$\mathbb{P}(c \mid a,b)$
a_1	b_1	c_1	0
a_1	b_1	c_2	0.5
a_1	b_1	c_3	0.5
a_1	b_2	c_1	0.2
a_1	b_2	c_2	0.3
a_1	b_2	c_3	0.5
a_2	b_1	c_1	0
a_2	b_1	c_2	0
a_2	b_1	c_3	1
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3
a_2	b_2	c_3	0.5

- ENC1 will generate 9 parameter variables
- **Group parameters that are equal and use a unique parameter variable by group**

$$a_1 \land b_1 \land c_2 \rightarrow \theta_{c1}$$
$$a_1 \land b_1 \land c_3 \rightarrow \theta_{c1}$$
$$a_1 \land b_2 \land c_3 \rightarrow \theta_{c1}$$
$$a_2 \land b_2 \land c_3 \rightarrow \theta_{c1}$$

with $w(\theta_{c1}) = 0.5$

Decomposability

- Model counters typically **look for dependent components**
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c \mid a, b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d \mid b, e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$$
\begin{array}{llll}
a_1 \wedge b_1 \wedge c_1 \to \theta_{c1} & \neg a_2 \vee \neg b_1 \vee \neg c_1 & b_1 \wedge e_1 \wedge d_1 \to \theta_{d5} & b_2 \wedge e_1 \wedge d_1 \to \theta_{d7} \\
a_1 \wedge b_1 \wedge c_2 \to \theta_{c1} & a_2 \wedge b_1 \wedge c_2 \to \theta_{c2} & b_1 \wedge e_1 \wedge d_2 \to \theta_{d6} & b_2 \wedge e_1 \wedge d_2 \to \theta_{d8} \\
a_1 \wedge b_2 \wedge c_1 \to \theta_{c1} & a_2 \wedge b_2 \wedge c_1 \to \theta_{c3} & b_1 \wedge e_2 \wedge d_1 \to \theta_{d7} & b_2 \wedge e_2 \wedge d_1 \to \theta_{d5} \\
a_1 \wedge b_2 \wedge c_2 \to \theta_{c1} & a_2 \wedge b_2 \wedge c_2 \to \theta_{c4} & b_1 \wedge e_2 \wedge d_2 \to \theta_{d8} & b_2 \wedge e_2 \wedge d_2 \to \theta_{d6}
\end{array}
$$

Decomposability

- Model counters typically **look for dependent components**
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c \mid a, b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d \mid b, e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$$\begin{array}{llll}
a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1} & \neg a_2 \vee \neg b_1 \vee \neg c_1 & b_1 \wedge e_1 \wedge d_1 \rightarrow \theta_{d5} & b_2 \wedge e_1 \wedge d_1 \rightarrow \theta_{d7} \\
a_1 \wedge b_1 \wedge c_2 \rightarrow \theta_{c1} & a_2 \wedge b_1 \wedge c_2 \rightarrow \theta_{c2} & b_1 \wedge e_1 \wedge d_2 \rightarrow \theta_{d6} & b_2 \wedge e_1 \wedge d_2 \rightarrow \theta_{d8} \\
a_1 \wedge b_2 \wedge c_1 \rightarrow \theta_{c1} & a_2 \wedge b_2 \wedge c_1 \rightarrow \theta_{c3} & b_1 \wedge e_2 \wedge d_1 \rightarrow \theta_{d7} & b_2 \wedge e_2 \wedge d_1 \rightarrow \theta_{d5} \\
a_1 \wedge b_2 \wedge c_2 \rightarrow \theta_{c1} & a_2 \wedge b_2 \wedge c_2 \rightarrow \theta_{c4} & b_1 \wedge e_2 \wedge d_2 \rightarrow \theta_{d8} & b_2 \wedge e_2 \wedge d_2 \rightarrow \theta_{d6}
\end{array}$$

- **Computing implicants** before translating into CNF **promotes the identification of independent components**

Decomposability

- Model counters typically **look for dependent components**
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c\|a,b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d\|b,e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1}$ $\quad \neg a_2 \vee \neg b_1 \vee \neg c_1$ $\quad b_1 \wedge e_1 \wedge d_1 \rightarrow \theta_{d5}$ $\quad b_2 \wedge e_1 \wedge d_1 \rightarrow \theta_{d7}$

$a_1 \wedge b_1 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_1 \wedge c_2 \rightarrow \theta_{c2}$ $\quad b_1 \wedge e_1 \wedge d_2 \rightarrow \theta_{d6}$ $\quad b_2 \wedge e_1 \wedge d_2 \rightarrow \theta_{d8}$

$a_1 \wedge b_2 \wedge c_1 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_1 \rightarrow \theta_{c3}$ $\quad b_1 \wedge e_2 \wedge d_1 \rightarrow \theta_{d7}$ $\quad b_2 \wedge e_2 \wedge d_1 \rightarrow \theta_{d5}$

$a_1 \wedge b_2 \wedge c_2 \rightarrow \theta_{c1}$ $\quad a_2 \wedge b_2 \wedge c_2 \rightarrow \theta_{c4}$ $\quad b_1 \wedge e_2 \wedge d_2 \rightarrow \theta_{d8}$ $\quad b_2 \wedge e_2 \wedge d_2 \rightarrow \theta_{d6}$

- **Computing implicants** before translating into CNF **promotes the identification of independent components**

Decomposability

- Model counters typically **look for dependent components**
- The translation into CNF may obfuscate the recognition of independent components

A	B	C	$\mathbb{P}(c \mid a,b)$
a_1	b_1	c_1	0.25
a_1	b_1	c_2	0.25
a_1	b_2	c_1	0.25
a_1	b_2	c_2	0.25
a_2	b_1	c_1	0
a_2	b_1	c_2	0.5
a_2	b_2	c_1	0.2
a_2	b_2	c_2	0.3

B	E	D	$\mathbb{P}(d \mid b,e)$
b_1	e_1	d_1	0.2
b_1	e_1	d_2	0.3
b_1	e_2	d_1	0.1
b_1	e_2	d_2	0.4
b_2	e_1	d_1	0.1
b_2	e_1	d_2	0.4
b_2	e_2	d_1	0.2
b_2	e_2	d_2	0.3

$a_1 \rightarrow \theta_{c1}$

$\neg a_2 \vee \neg b_1 \vee \neg c_1$
$a_2 \wedge b_1 \wedge c_2 \rightarrow \theta_{c2}$
$a_2 \wedge b_2 \wedge c_1 \rightarrow \theta_{c3}$
$a_2 \wedge b_2 \wedge c_2 \rightarrow \theta_{c4}$

$b_1 \wedge e_1 \wedge d_1 \rightarrow \theta_{d5}$
$b_1 \wedge e_1 \wedge d_2 \rightarrow \theta_{d6}$
$b_1 \wedge e_2 \wedge d_1 \rightarrow \theta_{d7}$
$b_1 \wedge e_2 \wedge d_2 \rightarrow \theta_{d8}$

$b_2 \wedge e_1 \wedge d_1 \rightarrow \theta_{d7}$
$b_2 \wedge e_1 \wedge d_2 \rightarrow \theta_{d8}$
$b_2 \wedge e_2 \wedge d_1 \rightarrow \theta_{d5}$
$b_2 \wedge e_2 \wedge d_2 \rightarrow \theta_{d6}$

- **Computing implicants** before translating into CNF **promotes the identification of independent components**

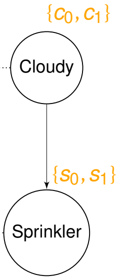
Chavira and Darwiche's Translation

$\top \to \theta_{c1}$

C	$\mathbb{P}(C)$
0	0.5
1	0.5

θ_{c1}

$c_0 \vee c_1$
$\overline{c_0} \vee \overline{c_1}$



$c_0 \to \theta_{s1}$
$s_0 \wedge c_1 \to \theta_{s3}$
$s_1 \wedge c_1 \to \theta_{s4}$

S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	θ_{s1}
1	0	0.5	
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}

$s_0 \vee s_1$
$\overline{s_0} \vee \overline{s_1}$

A more compact encoding of tables:

- Group rows with the same probability, and compress them (into a prime DNF);
- Use implications instead of equivalences.

Our Results: An Improved Translation

Two key ideas:

- Use a **log-encoding** of indicator variables
- Use a **scaling factor** w_0 which implicitly stores one group per table

Log Encoding

- In the log encoding domains are represented with $m = \lceil log_2(d) \rceil$ **propositional variables**
- Each of the 2^m combinations represents a possible assignment
- Introduce **less indicator** variables than the direct encoding

Log Encoding

- In the log encoding domains are represented with $m = \lceil log_2(d) \rceil$ **propositional variables**
- Each of the 2^m combinations represents a possible assignment
- Introduce **less indicator** variables than the direct encoding

green
blue
gray
purple
black
white

Log Encoding

- In the log encoding domains are represented with $m = \lceil log_2(d) \rceil$ **propositional variables**
- Each of the 2^m combinations represents a possible assignment
- Introduce **less indicator** variables than the direct encoding

green	0
blue	1
gray	2
purple	3
black	4
white	5

Log Encoding

- In the log encoding domains are represented with $m = \lceil log_2(d) \rceil$ **propositional variables**
- Each of the 2^m combinations represents a possible assignment
- Introduce **less indicator** variables than the direct encoding

3 propositional variables $\{c_2, c_1, c_0\}$

green	0
blue	1
gray	2
purple	3
black	4
white	5

Log Encoding

- In the log encoding domains are represented with $m = \lceil log_2(d) \rceil$ **propositional variables**
- Each of the 2^m combinations represents a possible assignment
- Introduce **less indicator** variables than the direct encoding

3 propositional variables $\{c_2, c_1, c_0\}$

green	0	$\neg c_2 \wedge \neg c_1 \wedge \neg c_0$
blue	1	$\neg c_2 \wedge \neg c_1 \wedge c_0$
gray	2	$\neg c_2 \wedge c_1 \wedge \neg c_0$
purple	3	$\neg c_2 \wedge c_1 \wedge c_0$
black	4	$c_2 \wedge \neg c_1 \wedge \neg c_0$
white	5	$c_2 \wedge \neg c_1 \wedge c_0$

Log Encoding

- In the log encoding domains are represented with $m = \lceil log_2(d) \rceil$ **propositional variables**
- Each of the 2^m combinations represents a possible assignment
- Introduce **less indicator** variables than the direct encoding

3 propositional variables $\{c_2, c_1, c_0\}$

green	0	$\neg c_2 \wedge \neg c_1 \wedge \neg c_0$
blue	1	$\neg c_2 \wedge \neg c_1 \wedge c_0$
gray	2	$\neg c_2 \wedge c_1 \wedge \neg c_0$
purple	3	$\neg c_2 \wedge c_1 \wedge c_0$
black	4	$c_2 \wedge \neg c_1 \wedge \neg c_0$
white	5	$c_2 \wedge \neg c_1 \wedge c_0$

- We need to exclude the values in excess with a **logarithmic number of clauses**

$$\neg c_2 \vee \neg c_1 \vee \neg c_0 \qquad \neg c_2 \vee \neg c_1 \vee c_0$$

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

A	B	C	$\mathbb{P}$(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

A	B	C	$\mathbb{P}$(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

A	B	C	$\mathbb{P}$(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

0.3

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

A	B	C	ℙ(C|A,B)
0	0	0	0.2
0	0	1	0.3
0	1	0	0.3
0	1	1	0.2
1	0	0	0
1	0	1	0
1	1	0	0.6
1	1	1	0.4

0.3

$\frac{0.2}{0.3}$

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

A	B	C	$\mathbb{P}$(C|A,B)	0.3
0	0	0	0.2	$\frac{0.2}{0.3}$
0	0	1	0.3	
0	1	0	0.3	
0	1	1	0.2	$\frac{0.2}{0.3}$
1	0	0	0	
1	0	1	0	
1	1	0	0.6	$\frac{0.6}{0.3}$
1	1	1	0.4	$\frac{0.4}{0.3}$

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

	A	B	C	$\mathbb{P}$(C|A,B)	0.3
θ_{c1}	0	0	0	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	0	0	1	0.3	
	0	1	0	0.3	
θ_{c1}	0	1	1	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	1	0	0	0	
	1	0	1	0	
θ_{c2}	1	1	0	0.6	$\frac{0.6}{0.3} = w(\theta_{c2})$
θ_{c3}	1	1	1	0.4	$\frac{0.4}{0.3} = w(\theta_{c3})$

$a_0 \wedge b_0 \wedge c_0 \rightarrow \theta_{c1}$ $\quad a_0 \wedge b_1 \wedge c_1 \rightarrow \theta_{c1}$

$a_1 \wedge b_1 \wedge c_0 \rightarrow \theta_{c2}$ $\quad a_1 \wedge b_1 \wedge c_1 \rightarrow \theta_{c3}$

Scaling Factor

- For each instantiation, **exactly one parameter variable will be assigned to true**
- Then, we can keep for each CPT R one parameter variable θ_R **implicit**

	A	B	C	$\mathbb{P}$(C|A,B)	0.3
θ_{c1}	0	0	0	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	0	0	1	0.3	
	0	1	0	0.3	
θ_{c1}	0	1	1	0.2	$\frac{0.2}{0.3} = w(\theta_{c1})$
	1	0	0	0	
	1	0	1	0	
θ_{c2}	1	1	0	0.6	$\frac{0.6}{0.3} = w(\theta_{c2})$
θ_{c3}	1	1	1	0.4	$\frac{0.4}{0.3} = w(\theta_{c3})$

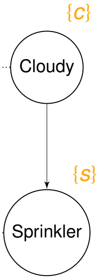
$$a_0 \land b_0 \land c_0 \rightarrow \theta_{c1} \quad a_0 \land b_1 \land c_1 \rightarrow \theta_{c1}$$
$$a_1 \land b_1 \land c_0 \rightarrow \theta_{c2} \quad a_1 \land b_1 \land c_1 \rightarrow \theta_{c3}$$

- To make the translation faithful we now assign a specific weight to the negative parameter literals: $w(\neg\theta_{ij}) = 1 - w(\theta_{ij})$

Running Example

C	$\mathbb{P}(C)$
0	0.5
1	0.5

S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	
1	0	0.5	
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}



no need!

no need!

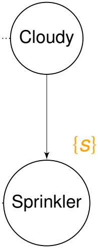
Running Example

$w_0 = \quad w_{s0} \quad \times \quad w_{c0}$

no need!

C	$\mathbb{P}(C)$
0	0.5
1	0.5

$\{c\}$



no need!

$\{s\}$

$\bar{s} \wedge c \rightarrow \theta_{s3}$
$s \wedge c \rightarrow \theta_{s4}$

S	C	$\mathbb{P}(S \mid C)$	
0	0	0.5	
1	0	0.5	
0	1	0.9	θ_{s3}
1	1	0.1	θ_{s4}

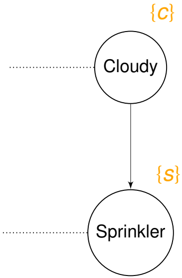
no need!

Running Example

$w_0 = \quad w_{s0} \quad \times \quad w_{c0}$

no need!

C	$\mathbb{P}(C)$
0	0.5
1	0.5



no need!

	S	C	$\mathbb{P}(S \mid C)$
	0	0	0.5
$\overline{s} \wedge c \rightarrow \theta_{s3}$	1	0	0.5
$s \wedge c \rightarrow \theta_{s4}$	0	1	0.9
	1	1	0.1

$w(\theta_{s3}) = 0.9/0.5$
$w(\theta_{s4}) = 0.1/0.5$

no need!

- The weighted model count of (F, w, w_0) becomes:

$$W(F) = w_0 \left(\sum_{\boldsymbol{v} \in \{0,1\}^n, \boldsymbol{v} \models F} W(\boldsymbol{v}) \right)$$

Our Results: An Improved Translation

Theoretical Properties

Let τ be the Chavira and Darwiche's translation, and let τ^* be our improvement. Then, for any **Bayesian network** N,

- the number of variables in $\tau^*(N)$ is smaller than the number of variables in $\tau(N)$,
- the size of $\tau^*(N)$ is smaller than the size of $\tau(N)$, and
- the translation τ^* is **faithful**:

$$W[\tau^*(\boldsymbol{x})] = \mathbb{P}_N(\boldsymbol{x}), \text{ for every variable instantiation } \boldsymbol{x}$$

Contrastingly, τ is not faithful (a DNNF minimization is required to achieve this).

Our Results: An Improved Translation

Theoretical Properties

Let τ be the Chavira and Darwiche's translation, and let τ^* be our improvement. Then, for any **Bayesian network** N,

- the number of variables in $\tau^*(N)$ is smaller than the number of variables in $\tau(N)$,
- the size of $\tau^*(N)$ is smaller than the size of $\tau(N)$, and
- the translation τ^* is **faithful**:

$$W[\tau^*(\boldsymbol{x})] = \mathbb{P}_N(\boldsymbol{x}), \text{ for every variable instantiation } \boldsymbol{x}$$

Contrastingly, τ is not faithful (a DNNF minimization is required to achieve this).

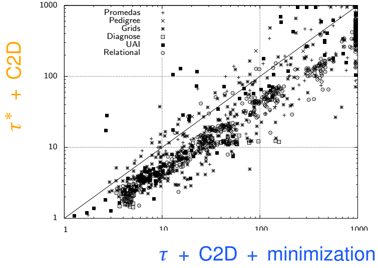
Note: The translations τ and τ^* can easily be extended to Markov networks, where $W[\tau(N)]$ captures the partition constant.

Our Results: An Improved Translation

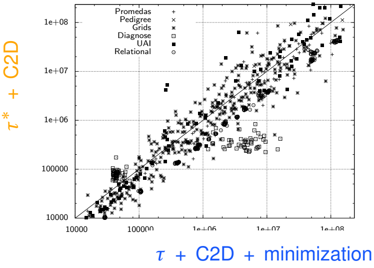
Experimental Setup

- 1452 graphical models
- 6 data sets: Diagnose (100), UAI (377), Grids (320), Pedigree (22), Promedas (238), Relational (395)
- Cluster of Quad-core Intel XEON X5550 with 32GiB of memory on Linux CentOS
- Time-out =900s (including the translation phase, and the minimization phase for τ)
- Memory-out = 8 GiB

Our Results: An Improved Translation



Compilation times

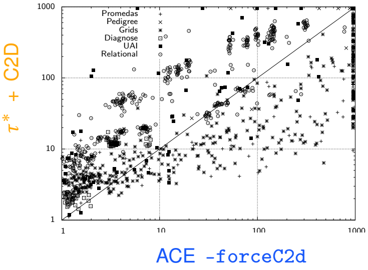


Sizes of compiled forms

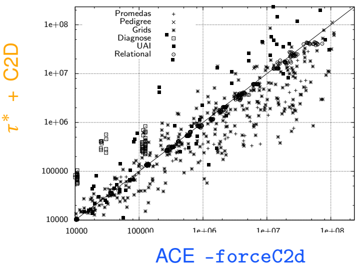
Experimental Results: τ^* versus τ

Evaluate the computational benefits offered by the new translation τ^* method, with respect to Chavira and Darwiche's method τ (+ minimization). Both methods are used upstream to C2D.

Our Results: An Improved Translation



Compilation times



Sizes of compiled forms

Experimental Results: τ^* + C2D versus ACE

Evaluate the computational benefits offered by the new translation τ^* (used upstream to C2D), with respect to ACE (used with `-forceC2d`), a compiler dedicated to graphical models.

Conclusions

Summary

- Weighted model counting is a promising approach for solving the probabilistic inference problem.
- We defined a new translation method based on two simple ideas;
  - τ^* is faithful
  - In practice τ^* + C2D proves typically better than τ + C2D + minimization,
  - In practice τ^* + C2D proves typically better than ACE as to the sizes of the compiled forms.

Conclusions

Summary

- Weighted model counting is a promising approach for solving the probabilistic inference problem.
- We defined a new translation method based on two simple ideas;
  - τ^* is faithful
  - In practice τ^* + C2D proves typically better than τ + C2D + minimization,
  - In practice τ^* + C2D proves typically better than ACE as to the sizes of the compiled forms.

Perspectives

Many questions remain open in reducing probabilistic inference to WMC. In particular, various translations can be devised:

- targetting other graphical models (relational, dynamic BNs, etc.),
- targetting other compiled forms (SDDs, etc.).

An Improved CNF Encoding Scheme for Probabilistic Inference

Jean-Marie Lagniez

CRIL, U. Artois & CNRS
France

UNIVERSITÉ D'ARTOIS